1. A computing system for performing stateful distributed computing comprising:

a client machine comprising a Client Runtime Environment (CRE); and

wherein said CRE is adapted to maintain state of an application by retrieving a first markup document of said application, creating and storing a first object oriented representation of information contained in said first markup document, wherein said first object oriented representation defines a first state of said application, retrieving a second markup document, creating and storing a second object oriented representation of information contained in said second markup document, and merging said first and said second object oriented representations thereby forming a new object oriented representation of information contained in said first or said second markup documents and wherein said new object oriented representation defines a new state of said application.

2. The system of claim 1 wherein said CRE further updates said new state of said application by retrieving one or more additional markup documents, creating and storing one or more additional object oriented representations of information contained in said one or more additional markup documents, respectively, and merging said one or more additional object oriented representations with said new object oriented representation thereby forming an updated state of said application.

3. The system of claim 1 wherein any of said object oriented representations comprise a structure selected from a group consisting of a user interface, a data structure, and business logic.

4. The system of claim 1 wherein said CRE further retrieves one or more procedural codes associated with any of said markup documents and executes said one or more procedural codes.

5. The system of claim 4 wherein said CRE further comprises an Application Program Interface (API) and said one or more procedural codes utilize said API for maintaining state of said application programmatically.

6. The system of claim 1 wherein said application state is maintained in a Document Object Model (DOM).

7. The system of claim 1 wherein any of said markup documents comprises an Extensible Markup Language (XML) format.

8. The system of claim 1 wherein any of said markup documents comprises a Hyper Text Markup Language (HTML) format.

9. The system of claim 4 wherein said one or more procedural codes are written in a programming language selected from a group consisting of C, C++, C#, Java, Javascript, VBScript, ActionScript, Visual Basic, and a proprietary programming language.

10. The system of claim 4 wherein said one or more procedural codes comprise binary format and said binary format is selected from a group consisting of .NET CLR, Java.class format, and Macromedia Flash binary format.

11. The system of claim 4 wherein said one or more procedural codes comprise text format and said text format is selected from a group consisting of HTML, XML, plain text, and compressed text.

12. The system of claim 1 wherein said merging comprises one or more operations selected from a group consisting of add, remove, insert, change, substitute, update, combine and combinations thereof.

13. The system of claim 12 wherein any of said object oriented representations comprise objects, object attributes, object attribute values, object hierarchical

relationships and combinations therefore and said one or more operations are applied to said objects, said object attributes, said object attribute values, said object hierarchical relationships and said combinations thereof.

14. The system of claim 4 wherein any of said one or more procedural codes and any of said markup documents are compiled and combined into a set of procedural code and said set of procedural code is retrieved and executed by said CRE.

15. The system of claim 1 wherein one or more of said markup documents are compiled into a procedural code and said procedural code is retrieved and executed by said CRE.

16. The system of claim 1 further comprising a real-time, bi-directional messaging system for sending and receiving messages between said client machine and a server over a network.

17. The system of claim 16 wherein said network comprises the World Wide Web (web).

18. The system of claim 16 wherein said network comprises a wireless network.

19. The system of claim 1 further comprising a web browser and said CRE runs inside said web browser.

20. The system of claim 1 further comprising a web browser and said CRE runs outside said web browser.

21. The system of claim 1 wherein said client machine is selected from a group consisting of a desktop computer, a laptop computer, a handheld device, and a smart phone.

22. The system of claim 1 further comprising one or more servers and said client machine is adapted to retrieve any of said markup documents from any of said one or more servers.

23. A computing method for maintaining state of an application in a client machine comprising:

retrieving a first markup document of said application;

creating a first object oriented representation of information contained in said first markup document and storing said first object oriented representation, wherein said first object oriented representation defines a first state of said application;

retrieving a second markup document of said application;

creating a second object oriented representation of information contained in said second markup document and storing said second object oriented representation; and

merging said first and said second object oriented representations thereby forming a new object oriented representation of information contained in said first or said second markup document and wherein said new object oriented representation defines a new state of said application.

24. The method of claim 23 further comprising updating said new state of said application by retrieving one or more additional markup documents, creating and storing one or more additional object oriented representations of information contained in said one or more additional markup documents, and merging said one or more additional object oriented representations with said new object oriented representation thereby forming an updated state of said application.

25. The method of claim 23 wherein any of said object oriented representations comprise a structure selected from a group consisting of a user interface, a data structure, and business logic.

26. The method of claim 23 further comprising retrieving one or more procedural codes associated with any of said markup documents and executing said one or more procedural codes.

27. The method of claim 26 wherein said one or more procedural codes utilizes an Application Program Interface (API) for maintaining state of said application programmatically.

28. The method of claim 23 wherein said application state is maintained in a Document Object Model (DOM).

29. The method of claim 23 wherein any of said markup documents comprises an Extensible Markup Language (XML) format.

30. The method of claim 23 wherein any of said markup documents comprises a Hyper Text Markup Language (HTML) format.

31. The method of claim 26 wherein said one or more procedural codes is written in a programming language selected from a group consisting of C, C++, C#, Java, Javascript, VBScript, ActionScript, Visual Basic, and a proprietary programming language.

32. The method of claim 26 wherein said one or more procedural codes comprise binary format and said binary format is selected from a group consisting of .NET CLR, Java.class format, and Macromedia Flash binary format.

33. The method of claim 26 wherein said one or more procedural codes comprise text format and said text format is selected from a group consisting of HTML, XML, plain text, and compressed text.

34. The method of claim 23 wherein said merging comprises executing one or more operations selected from a group consisting of add, remove, insert, change, substitute, update, combine and combinations thereof.

35. The method of claim 34 wherein any of said object oriented representations comprise objects, object attributes, object attribute values, object hierarchical relationships and combinations therefore and said one or more operations are applied to said objects, said object attributes, said object attribute values, said object hierarchical relationships and said combinations thereof.

36. The method of claim 26 wherein any of said one or more procedural codes and any of said markup documents are compiled and combined into a set of procedural code and said set of procedural code is retrieved and executed by said client machine.

37. The method of claim 23 wherein one or more of said markup documents are compiled into a procedural code and said procedural code is retrieved and executed by said client machine.

38. The method of claim 23 further comprising sending and receiving messages between said client machine and a server over a network via a real-time bidirectional messaging system.

39. The method of claim 23 wherein said client machine is selected from a group consisting of a desktop computer, a laptop computer, a handheld device, and a smart phone.

40. The method of claim 23 further comprising one or more servers and said client machine is adapted to retrieve any one of said markup documents from any of said one or more servers.

41. A method for developing an application adapted to run within a client machine wherein said client machine utilizes a Client Runtime Environment (CRE) for maintaining state of said application comprising:

defining user interface screens of said application as markup documents, respectively;

retrieving a first markup document by said CRE;

creating a first object representation of information contained in said first markup document by said CRE, wherein said first object oriented representation defines a first state of said application;

retrieving a second markup document by said CRE;

creating a second object oriented representation of information contained in said second markup document by said CRE;

merging said first and second object oriented representations by said CRE thereby forming a new object oriented representation of information contained in said first or said second markup document wherein said new object oriented representation defines a new state of said application.

42. The method of claim 41 further comprising updating said new state of said application by retrieving one or more additional markup documents, creating and storing one or more additional object oriented representations of information contained in said one or more additional markup documents, and merging said one or more additional object oriented representations with said new object oriented representation thereby forming an updated state of said application.

43. The method of claim 41 further comprising:

developing business logic associated with any of said user interface screens into one or more procedural codes; and

executing said one or more procedural codes by said CRE.

44. The method of claim 42 wherein said CRE comprises an Application Program Interface (API) and said method further comprises manipulating any of said markup documents and said application state via said API.

45. A method for deploying an application adapted to run within a client machine comprising:

deploying a Client Runtime Environment (CRE) in said client machine;

deploying said application on a central server, wherein said client machine is adapted to connect to said central server via a network and said central server is capable of serving requests from said CRE;

sending a first request from said CRE to said central server to download a first markup document of said application from said central server to said CRE;

creating and storing a first object oriented representation of said first markup document, wherein said first object oriented representation defines a first state of said application;

sending a second request from said CRE to said central server to download a second markup document of said application from said central server to said CRE ;

creating and storing a second object oriented representation of said second markup document; and

merging said first object oriented representation with said second object oriented representation thereby creating a new object oriented representation, wherein said new object oriented representation defines a new state of said application.

46. The method of claim 45 further comprising updating said new state of said application by retrieving one or more additional markup documents, creating and storing one or more additional object oriented representations of information contained in said one or more additional markup documents, and merging said one or more additional object oriented representations with said new object oriented representation thereby forming an updated state of said application.

47. The method of claim 45 further comprising downloading one or more procedural codes associated with any of said markup documents from said central server to said CRE and executing said one or more procedural codes by said CRE.

48. The method of claim 45 wherein any of said markup documents and one or more procedural codes are compiled and combined into a set of procedural code and said set of procedural code is retrieved and executed by said CRE.

49. The method of claim 47 wherein said one or more procedural codes comprise binary format and said binary format is selected from a group consisting of .NET CLR, Java.class format, and Macromedia Flash binary format.

50. The method of claim 47 wherein said one or more procedural codes comprise text format and said text format is selected from a group consisting of HTML, XML, plain text, and compressed text.

51. The method of claim 45 further comprising caching a client side application code in said client machine wherein said client side application code comprises said downloaded first and second markup documents, said new object oriented representation, one or more procedural codes and data downloaded from said central server to said client machine.

52. A method for deploying an application adapted to run within a client machine comprising:

deploying a Client Runtime Environment (CRE) in said client machine;

deploying said application on a central server, wherein said client machine is adapted to connect to said central server via a network and said central server is capable of serving requests from said CRE;

compiling and combining one or more markup documents and one or more procedural codes into an entity;

sending a request from said CRE to said central server to download said entity from said central server to said CRE;

retrieving a first information of a first markup document from said entity;

creating and storing a first object oriented representation of said first information, wherein said first object oriented representation defines a first state of said application;

retrieving second information of a second markup document from said entity ;

creating and storing a second object oriented representation of said second information; and

merging said first object oriented representation with said second object oriented representation thereby creating a new object oriented representation, wherein said new object oriented representation defines a new state of said application .

53. The method of claim 52 further comprising updating said new state of said application by retrieving one or more additional information of one or more additional markup documents, respectively, from said entity, creating and storing one or more additional object oriented representations of said one or more additional information, respectively, and merging said one or more additional object oriented representations with said new object oriented representation thereby forming an updated state of said application.

54. The method of claim 52 further comprising retrieving one or more procedural codes associated with any of said markup documents from said entity and executing said one or more procedural codes by said CRE

55. A distributed computing system for running an application over a network, wherein said application comprises a client side component and a server side component, said system comprising:

a client runtime environment (CRE) for running the client side component of the application and maintaining the client side application's state in a client side Document Object Model (DOM);

a server runtime environment (SRE) for running the server side component of the application and maintaining the server side application's state in a server side DOM; and

wherein said client side DOM is automatically synchronized with said server side DOM.

56. The system of claim 55 further comprising a real-time bi-directional messaging system for sending and receiving messages between the CRE and the SRE.

57. The system of claim 56 wherein said automatic synchronization between the client side DOM and the server side DOM is performed via said real-time bi-directional messaging system.

58. The system of claim 55 wherein said CRE runs inside a web browser.

59. The system of claim 55 wherein said SRE runs outside an application server.

60. The system of claim 55 wherein said SRE runs inside an application server.

61. The system of claim 55 wherein said client side DOM and said server side DOM comprise an Extensible Markup Language (XML) format.

62. The system of claim 55 wherein said CRE runs outside a web browser.

63. The system of claim 56 wherein said real-time bi-directional messaging system comprises HTTP messages.

64. The system of claim 56 wherein said real-time bidirectional messaging system establishes a persistent connection between said CRE and said SRE.

65. The system of claim 55 wherein when a connection between said CRE and said SRE over said network is interrupted the CRE and the SRE keep track of changes in the

client side DOM and the server side DOM, respectively, and synchronize them when said network connection is reestablished.

66. A distributed data storage system comprising:

a client side DOM for storing client side data;

a server side DOM for storing server side data;

a client side engine and a server side engine for synchronizing said client side DOM with said server side DOM and the reverse, respectively, over a network.

67. A method for performing 'server- push" of a plurality of messages from a server to a client machine comprising:

sending a normal HTTP request from said client machine to said server by opening an HTTP connection to said server;

accepting said HTTP connection by said server;

sending back to said client machine a response by said server wherein said response comprises an HTTP header instructing said client machine not to close said HTTP connection until a certain condition is met thereby maintaining said HTTP connection open; and

sending one or more of said plurality of messages to said client machine by said server via said open HTTP connection.

68. The method of claim 67 wherein said HTTP header comprises a "Content-type" header field indicating that said server response is a "multipart" mime-type response and said certain condition comprises an end of said "multipart" mime-type response.

69. The method of claim 67 wherein said HTTP header comprises a "Transfer-encoding" header field indicating that said server response is "chunked" and said certain condition comprises an end of said "chunked" response.

70. The method of claim 67 wherein said HTTP header comprises a "Content-length" header field indicating that said server response is a number that is bigger than a sum of

all content lengths of said plurality of messages, and said certain condition comprises a total number of bytes to be delivered equals or exceeds said number.

71. A communication system for performing "server-push" from a web application running inside an application server comprising:

a server module adapted to run inside said application server and to receive a request and to send a response to said request via a network connection;

a client machine adapted to send said request to said server module and to receive said response to said request via said network connection; and

wherein said server module performs 'server- push" of a plurality of messages to said client machine upon receipt of an HTTP request from said client machine and accepting an HTTP network connection opened by said client machine by sending back to said client machine a response comprising an HTTP header instructing said client machine not to close said HTTP network connection until a certain condition is met thereby maintaining said HTTP network connection open and then sending one or more of said plurality of messages to said client machine via said open HTTP network connection.

72. The system of claim 71 wherein said HTTP header comprises a "Content-type" header field indicating that said server response is a "multipart" mime-type response and said certain condition comprises an end of said "multipart" mime-type response.

73. The system of claim 71 wherein said HTTP header comprises a "Transfer-encoding" header field indicating that said server response is "chunked" and said certain condition comprises an end of said "chunked" response.

74. The system of claim 71 wherein said HTTP header comprises a "Content-length" header field indicating that said server response is a number that is bigger than a sum of all content lengths of said plurality of messages, and said certain condition comprises a total number of bytes to be delivered equals or exceeds said number.

75. The system of claim 71 wherein said application server is a J2EE application server and said web application is a Java web application.

76. The system of claim 71 wherein said application server is a .NET application server and said web application is a .NET web application.

77. The system of claim 71 wherein said server module is adapted to run behind said application server.

78. The system of claim 71 wherein said server module comprises an Application Program Interface (API) for sending messages to one or more client machines and said web application utilizes said API for performing "server-push" to said one or more client machines.